(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Veröffentlichungsnummer: **0 472 812 A1**

(12) EUROPÄISCHE PATENTANMELDUNG

(21) Anmeldenummer: **91107079.5**

(51) Int. Cl.5: **G06F 9/45**, G06F 9/445

(22) Anmeldetag: **02.05.91**

(30) Priorität: **28.08.90 CH 2793/90**

(43) Veröffentlichungstag der Anmeldung:
04.03.92 Patentblatt 92/10

(84) Benannte Vertragsstaaten:
CH DE GB LI SE

(71) Anmelder: **Landis & Gyr Betriebs AG**

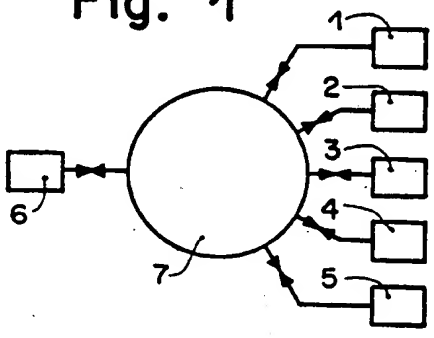
CH-6301 Zug(CH)

(72) Erfinder: **Wehrli, Herbert**
Seggenstrasse 8
CH-8865 Bilten(CH)
Erfinder: **Meyer, Mark**
Flachsacker 12
CH-6330 Cham(CH)

(54) Verfahren zum Aendern einer in einem Computer eines Gerätes abgespeicherten Maschinensprachenfassung eines ersten Programms in eine Maschinensprachenfassung eines durch mindestens eine Aenderung vom ersten Programm abgeleiteten zweiten Programms.

(57) Jeweils zum Zeitpunkt der Umwandlung einer Quellenfassung eines Programms in eine zugehörige Maschinensprachenfassung teilt ein "Compiler"-Programm die Quellenfassung in Segmente auf und ein "Linker"-Programm speichert Segmentinformationen in einer Zwischendatei. Das "Linker"-Programm liest während der Umwandlung des zweiten Programms die Segmentinformationen des ersten Programms und vergleicht sie mit den Segmentinformationen des zweiten Programms. Der Inhalt eines jeden Segmentes, der im zweiten Programm höchstens einen gleich grossen Speicherbedarf benötigt wie im ersten Programm, wird in beiden Maschinensprachenfassungen unter je einer gleichen Adresse abgespeichert. Der Inhalt eines jeden Segmentes, der in der Maschinensprachenfassung des zweiten Programms einen grösseren Speicherbedarf benötigt als im ersten Programm, wird unter Freigabe des bisher belegten Speicherbereichs zugunsten anderer Segmente, in einem freien Speicherbereich gespeichert. Ein Komparator/Generator-Programm vergleicht alle Bytes der beiden Maschinensprachenfassungen miteinander und erzeugt ein Unterschiedlichkeiten-Programm, welches nur mehr die für die beiden Maschinensprachenfassungen unterschiedlichen Bytes mit ihnen zugeordneten Adressen enthält und welches anschliessend dem Gerät (1 bis 5) zugeleitet wird, das mit einer Zentraleinheit (6) über ein Übertragungsnetz (7) verbunden ist.

Fig. 1



EP 0 472 812 A1

Die Erfindung bezieht sich auf ein verfahren zum Ändern einer in einem Computer eines Gerätes abgespeicherten Maschinensprachenfassung eines ersten Programms in eine Maschinensprachenfassung eines durch mindestens eine Änderung vom ersten Programm abgeleiteten zweiten Programms gemäss dem Oberbegriff des Anspruchs 1.

Das Verfahren wird zum verwalten des Änderungsdienstes von in technischen Geräten eingesetzten Computerprogrammen verwendet.

In Geräten und kleinen Systemen grösserer Fernübertragungs-Systemen werden heutzutage oft sehr leistungsfähige Computer eingesetzt, die wegen der Komplexität der von ihnen zu lösenden Aufgabe nicht mehr in einer maschinennahen Sprache, sondern in einer höheren, maschinenunabhängigen Programmiersprache, einer sogenannten Hochsprache ("high level language") programmiert werden, welche eine schnellere und sichere Programmierung, eine bessere Uebertragbarkeit des Programms auf einen anderen Computer und eine bessere Lesbarkeit des Programms gewährleistet.

Die Hochsprache ist z. B. PASCAL, MODULA 2, C, PORTAL, usw. Nachfolgend gilt die Annahme, dass PORTAL die verwendete Hochsprache ist, wobei das verfahren jedoch auch für die anderen Hochsprachen gültig ist.

Die als fernprogrammierbar angenommenen Geräte sind je mit einem Computer ausgerüstet und über einen oder mehrere Übertragungskanäle mit einer Zentraleinheit verbunden, die allen fernprogrammierbaren Geräten gemeinsam ist. Die Computer sind vorzugsweise Mikrocomputer.

Die in den Computern verwendeten Programme müssen in der Regel im Laufe der Zeit geändert werden, um sie neuen Anforderungen anzupassen, da z. B. für die alten Funktionen neue und bessere Algorithmen gefunden wurden oder den Gerätebenutzern neue, früher nicht vorgesehene Funktionen angeboten werden sollen. Während diese Aenderungen dem Gerätehersteller bei der Installation neuer Geräte in der Regel keine grosse Schwierigkeiten bereiten, stellen sie für die bereits im Einsatz befindliche Geräte in der Regel ein erhebliches Problem dar, da diese wegen ihrer grossen Zahl und wegen den grossen Entfernungen zwischen den Geräten in der Regel nicht vor Ort, sondern fern neu programmiert werden müssen, und dies in der Regel mehrmals während ihrer Lebensdauer.

Die Geräte sind z. B. Telefonkassierstationen eines öffentlichen Telefonnetzes, in denen z. B. bisher verwendete Taxationsverfahren geändert oder neue Dienstleistungen angeboten werden sollen, wie z. B. ein Abhol- oder Zubringerdienst bei Wahl einer Spezial-Nummer. Das öffentliche Telefonnetz beinhaltet in diesem Fall die Gesamtheit aller Uebertragungskanäle, mit denen die Geräte mit einer Telefonzentrale verbunden sind, die dann die erwähnte Zentraleinheit darstellt. Die Aenderungen des Computer-Programms einer Telefonkassierstation hat dann jeweils nach Möglichkeit fern über Telefonleitungen des öffentlichen Telefonnetzes zu erfolgen.

Bekannt ist das sogenannte "Patches"-Verfahren, mit dessen Hilfe in den Speichern der Geräte neue Programmteile zugeladen und/oder bestehende Programmteile überschrieben werden können, indem in der Maschinensprache, d. h. auf der Maschinencode-Ebene, und mit Kenntnis der verwendeten Machinencode-Adressen, die gewüschten Aenderungen von der Zentraleinheit zu den Geräten fernübertragen und anschliessend dort in deren Speichern nachgeladen werden. Dadurch gehen sehr schnell die Vorteile der Hochsprache-Programmierung verloren, wie z. B. die Qualität, die Unabhängigkeit vom der Maschinen-Sprache, der Dokumentationswert, usw..

Der Erfindung liegt die Aufgabe zugrunde, ein Verfahren der eingangs genannten Art zu verwirklichen, in dem der Benutzer keinen Zugriff zu Informationen bezüglich des erzeugten Maschinencodes und der verwendeten Maschinencode-Adressen benötigt und in dem

- einerseits vermieden wird, dass, insbesondere wenn eine grosse Anzahl von Geräten im Einsatz und/oder womöglich zu verschiedenen Zeiten und zu verschiedenen Zwecken von Programmänderungen betroffen sind, der Ueberblick über die verschiedenen, sich im Betrieb befindlichen Programmvarianten verloren geht, und
- anderseits vermieden wird, dass die Aktualität der Programmdokumentation verloren geht, wodurch verheerende Konsequenzen für die Programm-Erweiterungen und Programm-Aenderungen auf dem Niveau der Hochsprache entstehen können.

Die genannte Aufgabe wird erfindungsgemäss durch die im Kennzeichen des Anspruchs 1 angegebenen Merkmale gelöst. Vorteilhafte Ausgestaltungen der Erfindung ergeben sich aus den Unteransprüchen.

Ein Ausführungsbeispiel der Erfindung ist in der Zeichnung dargestellt und wird im folgenden näher beschrieben.

Es zeigen:

Fig. 1 ein Blockschaltbild eines Übertragungssystems,

Fig. 2 ein Blockschaltbild einer Computeranordnung,

Fig. 3 eine schematische Darstellung eines "Compiler/Linker"-Programms,

Fig. 4 eine schematische Darstellung eines erfindungsgemäss abgeänderten "Compiler/Linker"-Programms,

Fig. 5 ein Flussdiagramm eines erfindungsgemässen abgeänderten "Compiler/Linker"-Programms,

Fig. 6 eine Aufteilung einer Quellenfassung eines Programms in Segmente,

Fig. 7 ein Flussdiagramm eines Komparator/Generator-Programms und

Fig. 8 eine symbolische Darstellung einer Plazierung von Segmenten.

Gleiche Bezugszahlen bezeichnen in allen Figuren der Zeichnung gleiche Teile.

Das in der Fig. 1 dargestellte Übertragungssystem enthält mehrere, z. B. fünf Geräte 1, 2, 3, 4 und 5 sowie eine gemeinsame Zentraleinheit 6, die alle - nah oder fern - räumlich verteilt angeordnet sind. Die Zentraleinheit 6 ist über ein Übertragungsnetz 7 mit den Geräten 1 bis 5 verbunden, welches aus Drahtverbindungen, einem Funk-Netzwerk oder - vorzugsweise - einem Telefonnetz besteht. Übertragungssignale werden zwischen der Zentraleinheit 6 und den Geräten 1 bis 5 in beiden Übertragungsrichtungen übertragen. Falls die Übertragungssignale über grössere Entfernungen übertragen werden müssen, sind sie vorzugsweise modulierte Signale und werden dann als amplituden-, frequenz- oder phasenmodulierte Signale übertragen.

Die Geräte 1 bis 5 sind z. B. moderne Telefonkassier-Stationen, wobei dann die Zentraleinheit 6 eine gemeinsame Telefonzentrale und das Übertragungsnetz 7 ein Telefonnetz ist.

Die Geräte 1 bis 5 sowie die Zentraleinheit 6 enthalten je einen Computer 8, der jeweils in einer Computeranordnung 9 enthalten ist, deren prinzipieller Aufbau in der Fig. 2 vereinfacht dargestellt ist für den Fall, dass die Übertragungssignale modulierte Signale sind. Die Computeranordnung 9 besteht dann aus dem Computer 8 und einem Modem 10 (Modulator/Demodulator), welche in der angegebenen Reihenfolge in Kaskade geschaltet sind, wobei ein nicht mit dem Computer 8 verbundener erster Anschluss des Modems 10 einen Eingang/Ausgang der Computeranordnung 9 bildet und - nicht dargestellt- mit dem Übertragungsnetz 7 verbunden ist.

Der Computer 8 enthält seinerseits einen Zentralprozessor 11 (CPU: Central Processor Unit), einen Schreib/Lese-Speicher 12, einen Festwertspeicher 13 und einen Serie/Parallel-Wandler 14, die alle über Busverbindungen 15 miteinander verbunden sind. Der Schreib/Lese-Speicher 12 ist z. B. ein RAM (Random Access Memory) oder ein EEPROM (Electrically Erasable Programmable Read Only Memory), während der Festwertspeicher 13 z. B. ein ROM (Read Only Memory) oder ein PROM (Programmable Read Only Memory) ist. Mit Ausnahme des Busanschlusses des Festwertspeichers 13 werden alle Busverbindungen 15 in beiden Übertragungsrichtungen betrieben, während der Busanschluss des Festwertspeichers 13 nur in eine vom Festwertspeicher 13 wegweisende Übertragungsrichtung betrieben wird. Ein serieller Eingang/Ausgang des Serie/Parallel-Wandlers 14 bildet einen seriellen Eingang/Ausgang des Computers 8 und ist zwecks Bildung der Kaskadenschaltung mit einem zweiten Anschluss des Modems 10 verbunden, während ein Parallel-Eingang/Ausgang des Serie/Parallel-Wandlers 14 an die Busverbindungen 15 angeschlossen ist. Im Gegensatz zum beschriebenen Beispiel kann die Schnittstelle zwischen der Zentraleinheit 6 und den Geräten 1 bis 5 auch, vor allem bei relativ kurzen Übertragungsentfernungen, eine Parallel-Schnittstelle sein.

Im Computer 8 eines jeden Gerätes 1 bis 5 ist eine Maschinensprachenfassung X_1 eines ersten Programms Y_1 abgespeichert, welche mittels des erfindungsgemässen Verfahrens von der Zentraleinheit 6 aus in eine Maschinsprachenfassung X_2 eines abgeänderten zweiten Programm Y_2 umzuformen ist. Die beiden Maschinensprachenfassungen X_1 und X_2 sind in einer gleichen Maschinensprache abgefasst und ihnen entspricht jeweils im Computer 8 der Zentraleinheit 6 ein zugehöriges erstes beziehungsweise zweites Quellenprogramm Q_1 bzw. Q_2. Die beiden Quellenprogramme Q_1 und Q_2 sind in einer gleichen Hochsprache abgefasst und werden jeweils zu unterschiedlichen Zeitpunkten im Computer 8 der Zentraleinheit 6 mittels eines "Compiler/Linker-Programms" desselben in die Maschinensprachenfassung X_1 bzw. X_2des zugehörigen ersten beziehungsweise zweiten Programms Y_1 bzw. Y_2 umgewandelt. Das erste Programm Y_1 wird auch oft Vater-Programm und das zweite Programm Y_2 auch oft Sohn-Programm genannt.

Zu Beginn, bei der Installation der Zentraleinheit 6 und gewisser Geräte der Geräte 1 bis 5, ist ein als Basis-Programm Y_0 bezeichnetes Vater-Programm gültig, dessen Maschinensprachenfassung X_0 im Computers 8 der Zentraleinheit 6 unverlierbar gespeichert ist. Eine Kopie der Maschinensprachenfassung X_0 des Bais-Programms Y_0 ist ihrerseits unverlierbar im Computer 8 der betreffenden Geräte gespeichert.

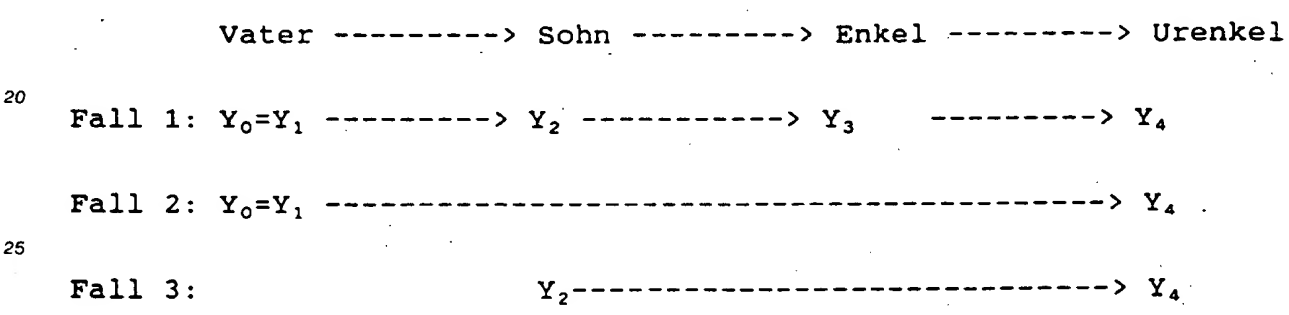
Kurz nach einer ersten Betriebaufnahme wird die Maschinensprachenfassung X_0 des Basis-Programms Y_0 in den Schreib/Lese-Speicher 12 der Zentraleinheit 6 und der gleichzeitig installierten Geräte übernommen und stellt dort eine Art erstes Vater-Programm dar. Im Laufe der Zeit werden neue zusätzliche Geräte installiert, die ein gleiches oder ein anderes Basis-Programm besitzen. Das andere Basis-Programm ist dabei in der Regel ein vom zuerst installierten Basis-Programm Y_0 mittels Änderungen abgeleitetes

Programm und ist eine Kopie der Maschinensprachenfassung eines entsprechenden, im Computer 8 der Zentraleinheit 6 abgespeicherten Vater- bzw. Sohn-Programms. Das gleiche oder andere Basis-Programm der zusätzlich installierten Geräten stellt für diese dann jeweils eine Art erstes Vater-Programm dar.

Im Laufe der Zeit können die Vater-Programme Y_1 der diversen Geräte 1 bis 5 von der Zentraleinheit 6 aus zu verschiedenen Zeitpunkten geändert werden. Die Vater-Programme Y_1 werden dann durch für die verschiedenen Geräte 1 bis 5 unterschiedliche Sohn-Programme ersetzt, die alle vom ursprünglichen Basis-Programm Y_0 direkt oder auf Umwege über verschiedene Vater-Programme abgeleitet sind und für jedes Gerät ganze Änderungsketten darstellen können. In einer solchen Änderungskette, werden die zeitlich aufeinanderfolgenden Programme der Änderungskette oft als Vater-, Sohn-, Enkel- und Urenkel-Programme bezeichnet.

Dabei ist jedes Programm der Änderungskette jeweils ein Sohn-Programm irgendeines vorgängigen Programms, welches dann dessen Vater-Programm darstellt, während das Basis-Programm Y_0 eine Art Urvater-Programm des Übermittlungssystems darstellt.

Beispiele:



In den Fällen 1 und 2 ist das Vaterprogramm Y_1 jeweils gleich dem Basis-Programm Y_0 und stellt dort jeweils ein erstes Vaterprogramm dar, während im Fall 3 das Sohn-Programm Y_2 ein erstes Vaterprogramm darstellt, welches auf irgendeine nicht dargestellte Art vom Basis-Programm Y_0 abgeleitet wurde.

Im Fall 1 wird im Laufe der Zeit die ganze Änderungskette schrittweise durchlaufen und aus dem Vaterprogramm Y_1 zuerst das Sohn-Programm Y_2 erzeugt, dann aus dem letzteren ein Enkel-Programm Y_3 generiert und schliesslich aus dem Enkel-Programm Y_3 ein Urenkel-Programm Y_4 erzeugt, wobei jeweils ein Vorgänger-Programm Y_1, Y_2 oder Y_3 ein Vater-Programm für das unmittelbar nachfolgende Programm Y_2, Y_3 bzw. Y_4 darstellt, welches dann ein Sohn-Programm des Vorgänger-Programms Y_1, Y_2 bzw. Y_3 ist.

Im Fall 2 wird die ganze Änderungskette in einem Schritt durchlaufen und direkt aus dem Vater-Programm Y_1 das Urenkel-Programm Y_4 erzeugt.

Der Fall 3 gehört zu einem später installierten Gerät, dessen Basis-Programm bereits das Sohn-Programm Y_2 ist, aus dem dann in einem einzigen Schritt, unter Überspringung des Enkel-Programms Y_3, das Urenkel-Programm Y_4 erzeugt wird.

Obwohl schliesslich in allen Geräten am Ende ein Urenkel-Programm Y_4 abgespeichert ist, bedeutet dies nicht, dass die Urenkel-Programme Y_4 aller Geräte identisch sind, da deren Zusammensetzung von der Entstehungsgeschichte und die Anzahl Zwischenschritte der Änderungskette abhängig ist. Die verschiedenen Geräte 1 bis 5 führen in diesem Fall trotzdem eine gleiche Nutzfunktion aus, da alle Urenkel-Programme Y_4 der Geräte 1 bis 5 ein gleiches Ergebnis ergeben.

Zusammengefasst kann festgestellt werden, dass alle Programme Y_1, Y_2, Y_3 und Y_4 der Änderungskette, soweit sie nicht identisch sind mit dem Basis-Programm Y_0, irgend einmal in ihrem Leben ein Sohn-Programm waren, so dass sie alle, inklusive die entsprechenden Vater-Programme Y_1, in der "Software" des Computers 8 der Zentraleinheit 6 wie ein Sohn-Programm behandelt werden können. Nachfolgend wird das Vater-Programm Y_1 wieder erstes und das Sohn-Programm Y_2 wieder zweites Programm genannt.

Für die "Software"-Entwicklung auf einem üblichen Computer hat ein Entwickler in der Regel ein "Compiler/Linker"-Programm 16, welches z. B. ein in der Fig. 3 dargestelltes Zweipass-"Compiler"-Programm ist und aus einem eigentlichen "Compiler"-Programm 17 und einem diesem nachgeordneten "Linker"-Programm 18 besteht. Dabei ist ein Ausgang des eigentlichen "Compiler"-Programms 17 auf einen Eingang des "Linker"-Programms 18 geführt.

Bekanntlich ist ein "Compiler"-Programm ein Programm, welches die in einer problemorientierten

Hochsprache abgefassten Quellenanweisungen eines Quellenprogramms Q in Zielanweisungen einer maschinenorientierten Programmiersprache übersetzt, und ein "Linker"-Programm ein Programm, welches der Erzeugung einer ladbaren Form X eines Computer-Programms dient und die durch das "Compiler"-Programm unabhängig voneinander übersetzten Programmteile miteinander verknüpft.

Zur Durchführung des erfindungsgemässen Verfahrens wurde das an sich bekannte und in der Fig. 3 dargestellte "Compiler/Linker"-Programm 16 für den Computer 8 der Zentraleinheit 6 abgeändert in ein "Compiler/Linker"-Programm 19, welches in der Fig. 4 dargestellt ist. Das "Compiler/Linker"-Programm 19 besteht aus dem eigentlichen "Compiler"-Programm 17, einem diesem nachgeordneten abgeänderten "Linker"-Programm 20 sowie einem dem letzteren nachgeordneten Komparator/Generator-Programm 21. Dabei ist der Ausgang des eigentlichen "Compiler"-Programms 17 auf einen ersten Eingang und ein Ausgang einer ersten Zwischendatei Z_1 auf einen zweiten Eingang des abgeänderten "Linker"-Programms 20 geführt, dessen erster Ausgang auf einen Eingang des Komparator/Generator-Programms 21 und dessen zweiter Ausgang auf einen Eingang einer zweiten Zwischendatei Z_2 geführt ist.

Die Quellenfassung Q_1 bzw. Q_2 des ersten Programms Y_1 bzw. des zweiten Programms Y_2 wird jeweils zum Zeitpunkt seiner Umwandlung in die zugehörige Maschinensprachenfassung X_1 bzw. X_2 einem Eingang des "Compiler"-Programms 17 des "Compiler/Linker"-Programms 16 bzw. 19 zugeführt, während die entsprechende Maschinensprachenfassung X_1 bzw. X_2 jeweils am Ausgang des "Linker"-Programms 18 bzw. am ersten Ausgang des "Linker"-Programms 20 erscheint. Die beiden Quellenfassungen Q_1 und Q_2 sowie die beiden Maschinensprachenfassungen X_1 und X_2 werden dabei jeweils im Schreib/Lese-Speicher 12 des Computers 8 der Zentraleinheit 6 gespeichert.

Beim erfindungsgemässen Verfahren werden jeweils anlässlich der Umwandlung der Quellenfassung Q_2 des zweiten Programms Y_2 in die zugehörige Maschinensprachenfassung X_2 die beiden am ersten Ausgang des abgeänderten "Linker"-Programms 20 erscheinenden Maschinensprachenfassungen X_1 und X_2 dem Komparator/Generator-Programm 21 zugeführt, der dann aus den beiden Maschinensprachenfassungen X_1 und X_2 ein Unterschiedlichkeiten-Programm δX erzeugt, dessen Bytes im Schreib/Lese-Speicher 12 des Computers 8 der Zentraleinheit 6 unter ihren zugeordneten Adressen abgespeichert werden. Ein Unterschiedlichkeiten-Programm δX wird bei jeder Kompilation eines geänderten Programms Y_2 erzeugt und in einem File abgelegt.

Die Bytes des Unterschiedlichkeiten-Programms δX werden anschliessend mittels des vorhandenen Übertragungsverfahrens von der Zentraleinheit 6 zu den Geräten 1 bis 5 übertragen. Zum Beispiel, wenn die Bits der Bytes des Unterschiedlichkeiten-Programms δX parallel anstehen, dann werden diese anschliessend mittels des Serie/Parallel-Wandlers 14 des Computers 8 der Zentraleinheit 6 (siehe Fig. 2) in zeitserielle Bits umgewandelt und - ggf. nach ihrer Modulation im Modem 10 der Zentraleinheit 6 - bitweise zeitseriell zum Gerät 1, 2, 3, 4 oder 5 übertragen. In der Computeranordnung 9 des betreffenden Gerätes werden dann die empfangenen zeitseriellen Bits - ggf. nach ihrer Demodulation im Modem 10 des Gerätes - mittels des Serie/Parallel-Wandlers 14 des Gerätes wieder in parallele Bits umgewandelt. Die empfangenen Bytes werden dann schliesslich im Schreib/Lese-Speicher 12 des Gerätes 1 unter einer dem Byte zugehörigen und mitübertragenen Adresse abgespeichert. Die so im Computer 8 des Gerätes abgespeicherten Bytes des Unterschiedlichkeiten-Programms δX bilden zusammen mit den bereits dort vorhandenen, nicht geänderten abgespeicherten Bytes der Maschinensprachenfassung X_1 des ersten Programms Y_1 des Gerätes dessen Maschinensprachenfassung X_2 des zweiten Programms Y_2, welches aufgabengemäss von der Zentraleinheit 6 aus zu generieren war.

Das aus dem "Compiler"-Programm 17 und dem "Linker"-Programm 20 bestehende Programm führt jeweils zum Zeitpunkt der Umwandlung der Quellenfassung Q_2 des zweiten Programms Y_2 in die zugehörige Machinensprachenfassung X_2 des Zweiten Programms Y_2 zeitlich nacheinander folgende Arbeitsabläufe durch, die in Fig. 5 in Gestalt eines Flussdiagramms symbolisch dargestellt sind.

Das in der Fig. 5 dargestellte Flussdiagramm enthält drei Funktionsblöcke 22, 23 und 24, vier Entscheidungsblöcke 25, 26 27 und 28 sowie einen Funktionsblock 29, die alle in der angegebenen Reihenfolge in Kaskade geschaltet sind und in der angegebenen Reihenfolge Funktionen oder Entscheidungen ausführen, welche in der Fig. 5 durch die Buchstaben A, B, C, D, E, F, G und H dargestellt sind. Zusätzlich sind im Flussdiagramm noch die drei Funktionsblöcke 30, 31 und 32 vorhanden, die in der angegebenen Reihenfolge je eine Funktion ausführen, die in der Fig. 5 durch einen Buchstaben I, J bzw. K dargestellt ist. Die Ja-Ausgänge der Entscheidungsblöcke 25 bis 28 sind jeweils mit Y (Yes) und die Nein-Ausgänge mit N (No) bezeichnet.

Ein Ausgang des Funktionsblocks 22 ist auf einen Eingang des Funktionsblocks 23 geführt, dessen Ausgang mit einem Eingang des Funktionsblocks 24 verbunden ist. Der Ausgang des Funktionsblocks 24 ist auf ein Eingang des Entscheidungsblocks 25 geführt, dessen Ja-Ausgang Y einen Ausgang des Flussdiagramms bildet. Der Nein-Ausgang N des Entscheidungsblocks 25 ist mit einem Eingang des Entschei-

dungsblocks 26 verbunden, dessen Nein-Ausgang N seinerseits auf einen Eingang des Entscheidungsblocks 27 geführt ist, dessen Ja-Ausgang Y wiederum mit einem Eingang des Entscheidungsblocks 28 verbunden ist. Der Ja-Ausgang Y des letzteren ist seinerseits auf einen Eingang des Funktionsblocks 29 geführt. Der Nein-Ausgang N des Entscheidungsblocks 27 ist mit einem Eingang des Funktionsblocks 31 verbunden, dessen Ausgang auf den Eingang des Entscheidungsblocks 26 geführt ist. Der Ja-Ausgang Y des Entscheidungsblocks 26 und der Nein-Ausgang N des Entscheidungsblocks 28 sind miteinander und mit einem Eingang des Funktionsblocks 30 verbunden, dessen Ausgang sowie ein Ausgang des Funktionsblocks 29 ebenfalls miteinander verbunden sind und auf den Eingang des Funktionsblocks 32 geführt sind, dessen Ausgang mit dem Eingang des Entscheidungsblocks 25 verbunden ist.

Die Buchstaben A bis K haben in der Fig. 5 jeweils folgende Bedeutung:

- A: Aufbereiten der Quellenfassung Q_1 bzw. Q_2 des anstehenden Programms Y_1 bzw. Y_2.
- B: Lesen des Inhaltes der Zwischendatei Z_1.
- C: Wähle das erste Segment des ersten Programms Y_1 als anstehendes Segment des ersten Programms Y_1 und das erste Segment des zweiten Programms Y_2 als anstehendes Segment des zweiten Programms Y_2.
- D: Sind alle Segmente des zweiten Programms Y_2 geprüft worden?
- E: Sind alle Segmente des ersten Programms Y_1 geprüft worden?
- F: Ist der Name des anstehenden Segmentes des zweiten Programms Y_2 identisch mit dem Namen des Segmentes des ersten Programms Y_1?
- G: Ist der durch das anstehende Segment des zweiten Programms Y_2 benötigte Speicherbedarf kleiner oder gleich als derjenige der durch das zugehörige anstehende Segment des ersten Programms Y_1 benötigt wird?
- H: Plaziere das anstehende Segment des zweiten Programms Y_2 am gleichen Speicherort wie das zugehörige Segment des ersten Programms Y_1.
- I: Plaziere das anstehende Segment des zweiten Programms Y_2 an einem freien Speicherort unter Freigabe des Speicherortes, an dem im ersten Programm Y_1 das zugehörige anstehende Segment des ersten Programms Y_1 gespeichert war, und unter anschliessender Korrektur in den anderen Segmenten des zweiten Programms Y_2 der Referenzangaben bezüglich des anstehenden Segmentes des Programms 2.
- J: Wähle das nächste Segment des ersten Programms Y_1.
- K: Wähle das nächste Segment des zweiten Programms Y_2.

Das Aufbereiten der anstehenden Quellenfassung im Funktionsblock 22 gemäss A beinhaltet dabei folgende Verfahrensschritte:

a) Das "Compiler"-Programm 17 des "Compiler/Linker"-Programms 19 teilt die Quellenfassung Q_1 bzw. Q_2 des anstehenden Programms Y_1 bzw. Y_2 in Segmente auf, die einen direkten Zusammenhang mit dem Inhalt, wie z. B. Routinen, Moduldatenfelder, usw., des betreffenden Programms Y_1 bzw. Y_2 besitzen und denen je ein Segment in der zugehörigen Maschinensprachenfassung X_1 bzw. X_2 des betreffenden Programms Y_1 bzw. Y_2 entspricht, wobei die Segmente in nachfolgenden "Compiler"-Durchläufen jeweils eine unteilbare Einheit bilden.

In der Fig. 6 ist ein Beispiel einer Aufteilung einer Quellenfassung Q eines kleinen Programms in Segmente dargestellt, welches keine praktische Bedeutung hat und nur der Illustration einer Aufteilung einer Quellenfassung in Segmente dient. Die Quellenfassung Q des Programms enthält z. B. nur ein einziges Programm-Modul mit dem Titel "test". Das Programm-Modul "test" beginnt mit einer Formulierung "module test" und endet mit einer Formulierung "end test". Das Programm-Modul "test" und damit die dargestellte Quellenfassung Q des Programms ist in der Darstellung der Fig. 6 in vier Segmente 33, 34, 35 und 36 aufgeteilt, die zur besseren Sichtbarmachung umrahmt dargestellt sind. In der Praxis fehlt diese Umrahmung jedoch normalerweise. Das Segment 33 stellt ein Datensegment des Programm-Moduls "test" dar, in welchem die Werte von Variablen i, j und k als ganzzahlige ("integer") und die Werte der Variablen r, l und q als reale ("real") Zahlen definiert werden. Die Segmente 34 und 35 stellen je ein Codesegment einer Prozedur ("procedure") p_1 bzw. p_2 dar. Das Segment 36 ist ein Codesegment des Programm-Moduls "test", in dem der Variablen i ein Wert 5 zugewiesen wird.

b) Das "Compiler"-Programm 17 des "Compiler/Linker"-Programms 19 ordnet jedem der erwähnten Segmente einen eindeutigen Segmentnamen zu, der für beide Programme Y_1 und Y_2 identisch ist.

Die Segmente werden mit einem Namen versehen, der einen eindeutigen Zusammenhang mit dem Bezeichner in der Quellenfassung hat, z. B. Name als String, Referenz in der Namenstabelle, Hashcode, usw.

Ausserdem müssen je nach verwendeter Hochsprache noch zusätzliche Strukturinformationen, wie z. B. Quellenfile, Schachtelungstiefe, usw., dem Segmentname zugefügt werden, da in gewissen Sprachen,

wie z. B. PASCAL, für verschiedene Routinen gleiche Namen verwendet werden können.

c) Das "Compiler"-Programm 17 des "Compiler/Linker"-Programms 19 fasst alle Segmente des anstehenden Programms Y_1 bzw. Y_2 in diverse Sektionen zusammen, wie z. B. Routinencodes, Variablen, Konstanten, usw.

Z. B. enthält die Sektion "CODE" alle Codeteile von Prozeduren, Funktionen und Prozessen, jedoch kein Modul-Code. Desgleichen enthält die Sektion "DATA" alle Variablen der Module.

d) Das "Linker"-Programm 20 des "Compiler/Linker"-Programms 19 plaziert die Sektionen und damit natürlich auch alle in ihnen enthaltenen Segmente anhand einer Spezifikation, z. B. in einem physikalischen Speicher oder in einem bestimmten Speicherbereich des Computers 8 der Zentraleinheit 6.

Beispiele:

- Region ROM 8000..FFFF : Physikalischer Speicherbereich
- Locate ROM CODE ..... : Plaziere die Sektionen CODE .... im Speicherbereich ROM

e) Das "Linker"-Programm 20 des "Compiler/Linker"-Programms 19 speichert Segmentinformationen bezüglich der plazierten Segmente des anstehenden ersten beziehungsweise zweiten Programms Y_1 bzw. Y_2 in einer ersten beziehungsweise zweiten Zwischendatei Z_1 bzw. Z_2, welche dann bei nachträglichen Kompilationen eines Sohn-Programms zur Verfügung stehen.

Die Zwischendateien Z_1 und Z_2 enthalten für jedes Segment in der Regel folgende, für das erfindungsgemässe Verfahren wesentliche Segmentinformationen:

- ein Segmentname,
- eine Sektionszugehörigkeit,
- eine Startadresse des Segmentes im physikalischen Speicher,
- ein Speicherbedarf des Segmentes in Bytes und
- eine maximal mögliche Grösse des Segmentes zwecks Lösung von Optimierungsproblemen bei Segmentverschiebungen.

Das "Linker"-Programm 20 des "Compiler/Linker"-Programms 19 liest anschliessend im Funktionsblock 23 gemäss B den Inhalt der Zwischendatei Z_1 des bereits zu einem früheren Zeitpunkt behandelten und aufbereiteten ersten Programms Y_1.

Danach wird im Funktionsblock 24 gemäss C das "Linker"-Programm 20 des "Compiler/Linker"-Programms 19 das erste Segment des ersten Programms Y_1 und das erste Segment des zweiten Programms Y_2 als anstehende Segmente der beiden Programme Y_1 und Y_2 wählen Mit anderen Worten: Der Computer 8 der Zentraleinheit 6 wird anschliessend zuerst einmal die in den Zwischendateien Z_1 und Z_2 gespeicherten Segmentinformationen, insbesondere die Segmentnamen und der Speicherbedarf, des ersten Segmentes der beiden Programme Y_1 und Y_2 behandeln und miteinander vergleichen sowie vom Vergleichsresultat ausgehend das erste Segment des zweiten Programms Y_2 plazieren.

Das "Linker"-Programm 20 stellt im Entscheidungsblock 25 gemäss D zuerst einmal fest, ob bereits alle Segmente des zweiten Programms Y_2 behandelt wurden. Wenn ja, dann ist das "Linker"-Programm 20 beendet und der Computer 8 der Zentraleinheit 6 kann zum nachgeordneten und in der Fig. 5 nicht mehr dargestellten Komparator/Generator-Programm 21 übergehen.

Wenn nein, dann stellt das "Linker"-Programm 20 im Entscheidungsblock 26 gemäss E fest, ob bereits alle Segmente des ersten Programms Y_1 behandelt wurden. Wenn ja, dann ist das anstehende Segment des zweiten Programms Y_2 nicht im ersten Programm vorhanden gewesen, es stellt somit ein neues Segment dar und muss im Funktionsblock 30 gemäss I in einer freien Speicherstelle abgespeichert werden.

Wenn nein, dann vergleicht das "Linker"-Programm 20 im Entscheidungsblock 27 gemäss F den Segmentnamen des anstehenden Segmentes des ersten und des zweiten Programms Y_1 und Y_2 auf Identität miteinander.

Bei Nichtidentität beider Segmentnamen geht das "Linker"-Programm 20 zum Funktionsblock 31 und wählt das nächste Segment des ersten Programms Y_1 und vergleicht, falls noch nicht alle Segmente des ersten Programms Y_1 behandelt wurde, dessen Segmentname mit demjenigen des noch immer anstehenden ersten Segmentes des zweiten Programms Y_2. Bei erneuter Nichtidentität wird gemäss Funktionsblock 31 das nächste Segment des ersten Programms Y_1 gewählt und dessen Segmentname mit demjenigen des anstehenden Segmentes des zweiten Programms verglichen. Das geschieht so oft und so lange bis entweder alle Segmente des ersten Programms Y_1 behandelt wurden oder bis die Segmentnamen-Identität vorhanden ist. Im ersten Fall geht das Programm wie bereits erläutert zum Funktionsblock 30 und im zweiten Fall zum Entscheidungsblock 28.

Bei vorhandener Segmentnamen-Identität, klärt das "Linker"-Programm 20 im Entscheidungsblock 28 gemäss G ab, ob der vom anstehenden Segment des zweiten Programms Y_2 benötigte Speicherbedarf

gleich oder kleiner ist als derjenige, der vom zugehörigen anstehenden Segment des ersten Programms Y_1 benötigt wird. Wenn ja, dann ist am alten Speicherort genügend Speicherplatz für das Segment des zweiten Programms Y_2 vorhanden und das Segment kann dort plaziert werden, was im Funktionsblock 29 gemäss H auch geschieht. Wenn nein, dann ist am alten Speicherort nicht genügend Speicherplatz für das anstehende Segment vorhanden und dieses muss gemäss I im Funktionsblock 30 in freien Speicherstellen an einem neuen Ort, z. B. am Ende des bisher geltenden Programms Y_1, plaziert werden unter Freigabe des alten Speicherortes. Letzterer steht dann zur freien Verfügung für eine nachfolgende Speicherung geänderter Segmente des zweiten Programms Y_2, wie z. B. neuer Segmente oder Segmente, die einen höheren Speicherbedarf als das entsprechende Segment des ersten Programms Y_1, jedoch einen niedrigeren speicherbedarf benötigen, als im freigegebenen Speicherort vorhanden ist. Ausserdem werden anschliessend dann noch alle Referenzangaben, die sich in anderen Segmenten auf das betreffende anstehende Segment beziehen, wie z. B. Routinenaufrufe, korrigiert und auf den neuesten Stand gebracht, da dieses Segment ja jetzt eine neue, andere Stelle im Speicher und im Programm einnimmt.

Nach der Plazierung des ersten Segmentes des zweiten Programms Y_2 geht das "Linker"-Programm 20 dann zum Funktionsblock 32 und wählt gemäss K das nächste Segment des zweiten Programms Y_2 zur Behandlung, was auf die gleiche Art geschieht, wie beim ersten Segment.

Auf diese Weise werden mittels des "Linker"-Programms 20 zeitlich nacheinander alle Segmente des zweiten Programms Y_2 behandelt bis zu dem Augenblick, an dem der Entscheidungsblock 25 feststellt, dass alle Segmente des zweiten Programms Y_2 behandelt wurden und das Programm das "Compiler/Linker"-Programm 20 verlässt, um zum Komparator/Generator-Programm 21 über zu gehen (siehe Fig. 4).

Das "Linker"-Programm 20 sucht somit für jedes Segment des zweiten Programms Y_2 alle Segmente des ersten Programms Y_1 auf Segmentnamen-Identität ab und plaziert dann bei vorhandener Segmentnamen-Identität das anstehehende Segment des zweiten Programms Y_2 entsprechend seinem speicherbedarf. Dabei wird der Inhalt eines jeden Segmentes, der im zweiten Programm Y_2 einen grösseren Speicherbedarf benötigt als im ersten Programm Y_1, unter Freigabe des bisher belegten Speicherbereichs zugunsten des Inhaltes eines oder mehrerer anderer Segmente, in einem freien Speicherbereich des Computers 8 der Zentraleinheit 6 gespeichert, während alle andere Segmente des zweiten Programms Y_2, die im zweiten Programm Y_2 höchstens je einen gleich grossen Speicherbedarf benötigen wie im ersten Programm Y_1, am gleichen Ort gespeichert werden wie das zugehörige Segment des ersten Programms Y_1 im letzteren gespeichert war.

Die so plazierten Segmente des zweiten Programms Y_2 bilden dann zusammen die Maschinensprachenfassung X_2 des anstehenden Programms Y_2, welche somit vom "Linker"-Programm 20 des "Compiler/Linker"-Programms 19 erzeugt wurde, um unter anderem dem Komparator/Generator-Programm 21 zur Verfügung gestellt zu werden.

Der Ausgang des in der Fig. 5 dargestellten Flussdiagramms, d. h. der Ja-Ausgang des Entscheidungsblocks 25, ist auf einen Eingang des in der Fig. 7 dargestellten Flussdiagramms des Komparator/Generator-Programms 21 geführt, welches aus einem Funktionsblock 37, drei Entscheidungsblöcken 38, 39 und 40 sowie zwei Funktionsblöcken 41 und 42 besteht, die alle in der angegebenen Reihenfolge in Kaskade geschaltet sind und die in der angegebenen Reihenfolge Entscheidungen oder Funktionen ausführen, welche in der Fig. 7 durch die Buchstaben L, M, R, S, T und U dargestellt sind. Zusätzlich sind noch in der Fig. 7 die beiden Funktionsblöcke 43 und 44 vorhanden, die je eine Funktion ausführen, die in der Fig. 7 durch einen Buchstaben V bzw. W dargestellt ist. Die Ja-Ausgänge der Entscheidungsblöcke 38 bis 40 sind jeweils mit Y und die Nein-Ausgänge mit N bezeichnet. Ein Eingang des Funktionsblocks 37 bildet den Eingang des Flussdiagramms des Komparator/Generator-Programms 21, während der Ausgang des letzteren durch einen Ausgang des Funktionsblocks 44 gebildet wird.

Ein Ausgang des Funktionsblocks 37 ist auf einen Eingang des Entscheidungsblocks 38 geführt, dessen Nein-Ausgang N mit einem Eingang des Entscheidungsblocks 39 verbunden ist, dessen Nein-Ausgang N seinerseits auf einen Eingang des Entscheidungsblocks 40 geführt ist, dessen Ja-Ausgang Y wiederum mit einem Eingang des Funktionsblocks 41 verbunden ist. Der Nein-Ausgang N des Entscheidungsblocks 40 und der Ausgang des Funktionsblocks 41 sind miteinander und mit einem Eingang des Funktionsblocks 42 verbunden. Der Ja-Ausgang Y des Entscheidungsblocks 39 und der Ausgang des Funktionsblocks 42 sind miteinander und mit einem Eingang des Funktionsblocks 43 verbunden, dessen Ausgang auf den Eingang des Entscheidungsblocks 38 geführt ist. Der Ja-Ausgang Y des letzteren ist auf einen Eingang des Funktionsblocks 44 geführt.

Die Buchstaben L bis W haben in der Fig. 7 jeweils folgende Bedeutung:

L: Lese die Adresse und den Inhalt des ersten Bytes der Maschinensprachenfassung X_2 des zweiten Programms Y_2 sowie den Inhalt des zugehörigen Bytes der Maschinensprachenfassung

X_1 des ersten Programms Y_1.

M: Sind alle Bytes der Maschinensprachenfassung X_2 des zweiten Programms Y_2 geprüft worden?

R: Ist das anstehende Byte der Maschinensprachenfassung X_2 des zweiten Programms Y_2 identisch mit dem zugehörigen Byte der Maschinensprachenfassung X_1 des ersten Programms Y_1?

S: Ist die Differenz zwischen der Maschinensprachen-Adresse des anstehenden Bytes des zweiten Programms Y_2 und der Maschinen-Sprachen-Adresse des zuletzt im Unterschiedlichkeiten-Programm δX plazierten Bytes des zweiten Programms Y_2 grösser als Eins?

T: Plaziere die Maschinensprachen-Adresse des anstehenden Bytes des zweiten Programms Y_2 in die nächste freie Stelle des Unterschiedlichkeiten-Programms δX.

U: Plaziere das anstehende Byte der Maschinensprachenfassung X_2 des zweiten Programms Y_2 in die nächste freie Stelle des Unterschiedlichkeiten-Programms δX und speichere die Adresse, die dieses Byte in der Maschinensprachenfassung X_2 des zweiten Programms Y_2 besitzt, im Schreib/Lese-Speicher 12 des Computers 8 der Zentraleinheit 6 an einer zu diesem Zweck reservierten Stelle.

V: Lese die Adresse und den Inhalt des nächsten Bytes der Maschinensprachenfassung X_2 des zweiten Programms Y_2 sowie den Inhalt des zugehörigen Bytes der Maschinensprachenfassung X_1 des ersten Programms Y_1.

W: Lese das Unterschiedlichkeiten-Programm δX.

Das Komparator/Generator-Programm 21 behandelt gemäss der Funktionsblöcke 37 und 43 zeitlich nacheinander die Adressen und Inhalte aller Bytes aller Segmente der Maschinensprachenfassung X_2 des zweiten Programms Y_2. Es stellt im Entscheidungsblock 38 gemäss M zuerst einmal fest, ob bereits alle Bytes der Maschinensprachenfassung X_2 des zweiten Programms Y_2 behandelt wurden. Wenn ja, dann ist das Komparator/Generator-Programm 21 an sich beendet und der Inhalt des Unterschiedlichkeiten-Programms δX braucht nur mehr im Funktionsblock 44 gemäss W durch den Computer 8 der Zentraleinheit 6 gelesen und zu einem oder mehreren der Geräte 1 bis 5 übertragen zu werden.

Wenn nein, dann vergleicht das Komparator/Generator-Programm 21 im Entscheidungsblock 39 gemäss R den Inhalt des anstehenden Bytes der Maschinensprachenfassung X_2 des zweiten Programms Y_2 mit dem ebenfalls anstehenden Inhalt des zugehörigen Bytes der Maschinensprachenfassung X_1 des ersten Programms Y_1. Bei einer Identität beider Inhalte geht das Komparator/Generator-Programm 21 zum Funktionsblock 43 und gibt den Auftrag das nächste Byte der Maschinensprachenfassung X_2 des zweiten Programms Y_2 zu behandeln, weil das anstehende Byte unverändert geblieben ist und nicht weiter behandelt werden muss, da es von der bereits in den betroffenen Geräten 1 bis 5 vorhandenen Maschinensprachenfassung X_1 des ersten Programms Y_1 übernommen werden kann und daher nicht Gegenstand des Unterschiedlichkeiten-Programms δX zu sein braucht.

Bei einer nicht vorhandenen Identität beider Inhalte, klärt das Komparator/Generator-Programm 21 anschliessend im Entscheidungsblock 40 gemäss S ab, ob die Differenz zwischen der Adresse des anstehenden Bytes in der Machinensprachen-Fassung X_2 des zweiten Programms Y_2 und der Adresse in der Maschinensprachenfassung X_2 des zweiten Programms Y_2 des zuletzt im Unterschiedlichkeiten-Programm δX plazierten Bytes grösser als Eins ist.

- Wenn ja, dann besitzen zwei aufeinanderfolgend in das Unterschiedlichkeiten-Programm δX plazierte Bytes in der Maschinensprachenfassung X_2 des zweiten Programms Y_2 keine aufeinanderfolgende Adressen und bilden somit dort keinen Programmblock. Es ist somit ein Adressensprung vorhanden, der im Unterschiedlichkeiten-Programm δX dadurch berücksichtigt wird, dass im Funktionsblock 41 gemäss T die Adresse des anstehenden Bytes in die nächste freie Stelle des Unterschiedlichkeiten-Programms δX plaziert wird. Anschliessend wird dann im Funktionsblock 42 gemäss U das anstehende Byte der Maschinensprachenfassung X_2 des zweiten Programms Y_2 an der nächsten freien Stelle in das Unterschiedlichkeiten-Programm δX plaziert.
- Wenn nein, dann besitzen zwei aufeinanderfolgend in das Unterschiedlichkeiten-Programm δX plazierte Bytes in der Maschinensprachenfassung X_2 des zweiten Programms Y_2 aufeinanderfolgende Adressen und bilden somit dort einen Programmblock. Es ist somit kein Adressensprung vorhanden, der im Unterschiedlichkeiten-Programm δX zu berücksichtigen wäre. Der Funktionsblock 41 kann somit übersprungen und im Funktionsblock 42 gemäss U direkt der Inhalt des anstehenden Bytes der Maschinensprachenfassung X_2 des zweiten Programms Y_2 an der nächsten freien Stelle in das Unterschiedlichkeiten-Programm δX plaziert werden.

Nach der Plazierung des anstehenden Bytes der Maschinensprachen-Fassung X_2 des zweiten Programms Y_2 in das Unterschiedlichkeiten-Programm δX geht das Komparator/Generator-Programm 21 zum Funktionsblock 43 und gibt den Auftrag das nächste Byte zu behandeln. Das Komparator/Generator-Programm 21 vergleicht somit alle Bytes aller Segmente der beiden Maschinensprachenfassungen X_1 und

X_2 miteinander und erzeugt, vom Vergleichsresultat ausgehend, das Unterschiedlichkeiten-Programm δX, welches nur mehr die für die beiden Maschinensprachenfassungen X_1 und X_2 unterschiedlichen Bytes mit ihnen zugeordneten Adressen enthält, wobei zum Einsparen von Speicherplätzen im Unterschiedlichkeiten-Programm δX und zum Einsparen von Übertragungszeit zwischen der zentralen Steuereinheit 6 und den Geräten 1 bis 5, nur das absolute Minimum dieser Adressen in das Unterschiedlichkeiten-Programm δX übernommen wird. Mit anderen Worten: Aufeinanderfolgende Bytes der Maschinensprachenfassung X_2 werden, falls sie in das Unterschiedlichkeiten-Programm δX übernommen werden, als Programmblock mit einer einzigen Adresse behandelt und ihnen die Adresse ihres ersten Bytes zugeordnet, während nicht aufeinanderfolgende übernommene Bytes der Maschinensprachenfassung X_2 je eine getrennte Adresse beibehalten und mit dieser in das Unterschiedlichkeiten-Programm δX übernommen werden.

Nachdem alle Bytes aller Segmente der Maschinensprachenfassung X_2 geprüft und ggf. in das Unterschiedlichkeiten-Programm δX übernommen, ist letzteres erstellt und abgespeichert. Seine Bytes können dann im Funktionsblock 44 gemäss W gelesen und nach Aufbereitung zu einem oder mehreren der Geräte 1 bis 5 übertragen werden.

In der Fig. 8 sind die Maschinensprachenfassungen X_1 und X_2 der beiden Programme Y_1 und Y_2 sowie das Unterschiedlichkeiten-Programm δX symbolisch wiedergegeben, indem ihre einzelne Segmente jeweils als schraffierte Blöcke untereinander dargestellt sind in einer Reihenfolge, in der sie auch abgespeichert sind. Dabei sind zur besseren Sichtbarmachung aufeinanderfolgende Segmente abwechselnd unterschiedlich schraffiert dargestellt. Nicht schraffierte Blöcke sind in der Maschinensprachenfassung X_2 leer bzw. im Unterschiedlichkeiten-Programm δX leer oder mit einer Adresse a1, a2, a3 oder a4 belegt. Die Machinensprachen-Fassung X_1 des ersten Programms Y_1 besteht aus den Segmenten 45 bis 61, während die Maschinensprachenfassung X_2 des zweiten Programms Y_2 am gleichen Ort wie in der Maschinensprachenfassung X_1 die Segmente 45 bis 53 sowie 55 bis 61 besitzt. Da das Segment 54 von X_1 in X_2 mehr Speicherplatz benötigt als in X_1, wurde sein Speicherplatz 54 in X_2 leer gelassen (siehe weisser Block 54 in X_2) und das geänderte Segment 54 als Segment 62 direkt anschliessend nach dem Segment 61 an einer freien Speicherstelle in X_2 untergebracht. Das Segment 63 von X_2 ist dagegen ein neues Segment, welches in X_1 kein Äquivalent und demnach auch keine Speicherstelle besitzt, und direkt anschliessend nach dem Segment 62 am Ende von X_2 untergebracht. Die abgeänderten Segmente 48 und 58 benötigen in X_2 weniger Speicherplatz als in X_1, so dass sie in X_2 am gleichen Ort wie in X_1 plaziert werden können, der in X_2 jedoch jeweils nur teilweise belegt ist (siehe in X_2 die weissen Blöcke direkt anschliessend an den schraffierten Blöcken 48 und 58). Nachfolgend gilt aus Gründen der zeichnerischen Einfachheit, dass alle Bytes der Segmente 48, 54 und 58 geändert wurden.

Für alle anderen Segmenten 45 bis 47, 49 bis 53, 55 bis 57 und 59 bis 61 gilt die Annahme, dass sie unverändert geblieben sind und sie somit in X_2 vollständig den gleichen Ort belegen wie in X_1. Diese unveränderten Segmente brauchen nicht in das Unterschiedlichkeiten-Programm δX übernommen zu werden und können daher nachfolgend vergessen werden. Die geänderten Segmente 48, 54 und 58 sowie die neuen Segmente 62 und 63 sind in X_2 in der angegebenen Reihenfolge untergebracht, wobei das Segment 54 von X_2 leer ist und das Segment 62 von X_2 dem geänderten Segment 54 von X_1 entspricht. Die geänderten Segmente 48, 54, 58, 62 und 63 von X_2 werden in dieser Reihenfolge auch in das Unterschiedlichkeiten-Programm δX übernommen, wobei die Segmente 62 und 63, da aufeinanderfolgend, im Unterschiedlichkeiten-Programm δX einen Block bilden, dem nur eine einzige Adresse a4 benötigt. Die drei Segmente 48, 54 und 58 sowie der Segmentblock 62;63 sind in X_2 durch andere Segmente voneinander getrennt und benötigen daher im Unterschiedlichkeiten-Programm δX je eine Adresse a1, a2, a3 bzw. a4. Das Unterschiedlichkeiten-Programm δX besteht dann in der angegebenen Reihenfolge aus den Blöcken a1, 48, a2, 54, a3, 58, a4, 62, 63.

Patentansprüche

1. Verfahren zum Ändern einer in einem Computer (8) eines Gerätes (1, 2, 3, 4 oder 5) abgespeicherten Maschinensprachenfassung (X_1) eines ersten Programms (Y_1) in eine Maschinensprachenfassung (X_2) eines durch mindestens eine Änderung vom ersten Programm (Y_1) abgeleiteten zweiten Programms (Y_2), wobei jeweils zu unterschiedlichen Zeitpunkten eine zugehörige erste beziehungsweise zweite Quellenfassung (Q_1 bzw. Q_2) der beiden Programme (Y_1, Y_2) in einem Computer (8) einer Zentraleinheit (6) mittels eines "Compiler/Linker"-Programms (19) in die zugehörige erste beziehungsweise zweite Maschinenensprachenfassung (X_1 bzw. X_2) umgewandelt wird,

   dadurch gekennzeichnet,

- dass jeweils zum Zeitpunkt der Umwandlung einer Quellenfassung (Q1 bzw. Q_2) in die zugehörige Machinensprachenfassung (X_1 bzw. X_2) im Computer (8) der Zentraleinheit (6)
  - ein "Compiler"-Programm (17) des "Compiler/Linker"-Programms (19) die betreffende Quellenfassung (Q_1 bzw. Q_2) in Segmente aufteilt, die einen direkten Zusammenhang mit dem Inhalt der Quellenfassung (Q_1 bzw. Q_2) besitzen und denen je ein Segment in der zugehörigen Maschinensprachenfassung (X_1 bzw. X_2) entspricht, wobei die Segmente in nachfolgenden "Compiler"-Durchläufen jeweils eine unteilbare Einheit bilden,
  - ein "Linker"-Programm (20) des "Compiler/Linker"-Programms (19) Segmentinformationen bezüglich der Segmente des betreffenden Programms (Y_1 bzw. Y_2) in einer Zwischendatei (Z_1 bzw. Z_2) speichert, und
- dass das "Linker"-Programm (20) während der Umwandlung der Quellenfassung (Q_2) des zweiten Programms (Y_2) in dessen Maschinensprachenfassung (X_2)
- die in einer ersten Zwischendatei (Z_1) gespeicherten Segmentinformationen des ersten Programms (Y_1) liest und mit den in einer zweiten Zwischendatei (Z_2) gespeicherten Segmentinformationen des zweiten Programms (Y_2) vergleicht,
- den Inhalt eines jeden Segmentes, der in der Maschinensprachenfassung (X_2) des zweiten Programms (Y_2) höchstens einen gleich grossen Speicherbedarf benötigt wie in derjenigen des ersten Programms (Y_1), in beiden Maschinensprachenfassungen (X_1, X_2) unter je einer gleichen Adresse im Computer (8) der Zentraleinheit (6) abspeichert, und
- den Inhalt eines jeden Segmentes, der in der Maschinensprachenfassung (X_2) des zweiten Programms (Y_2) einen grösseren Speicherbedarf benötigt als in derjenigen des ersten Programms (Y_1),
  - unter Freigabe des bisher belegten Speicherbereichs zugunsten des Inhaltes eines oder mehrerer anderer Segmente, in einem freien Speicherbereich des Computers (8) der Zentraleinheit (6) speichert sowie
  - Referenzangaben, die sich in anderen Segmenten auf das betreffende Segment beziehen, korrigiert und auf den neuesten Stand bringt,
- dass dem "Linker"-Programm (20) ein Komparator/Generator-Programm (21) nachgeordnet ist, welches
  - alle Bytes der beiden Maschinensprachenfassungen (X_1, X_2) miteinander vergleicht und
  - ein Unterschiedlichkeiten-Programm (δX) erzeugt, welches nur mehr die für die beiden Maschinensprachenfassungen (X_1, X_2) unterschiedlichen Bytes mit ihnen zugeordneten Adressen (a1, a2, a3, a4) enthält, und
- dass das Unterschiedlichkeiten-Programm (δX) dem Computer (8) des Gerätes (1, 2, 3, 4 oder 5) zugeleitet wird und seine Bytes dort unter ihren zugeordneten Adressen (a1, a2, a3, a4) abgespeichert werden, wobei sie zusammen mit den bereits vorhandenen, nicht geänderten Bytes der Maschinensprachenfassung (X_1) des ersten Programms (Y_1) des Gerätes (1, 2, 3, 4 oder 5) dessen Maschinensprachenfassung (X_2) des zweiten Programms (Y_2) bilden.

2. Verfahren nach Anspruch 1, dadurch gekennzeichnet, dass das "Compiler"-Programm (17) jedem der Segmente einen eindeutigen Segmentnamen zuordnet, der für beide Programme (Y_1, Y_2) identisch ist.

3. Verfahren nach Anspruch 1 oder 2, dadurch gekennzeichnet, dass die Segmentinformationen jeweils mindestens einen Segmentnamen, eine Startadresse des Segmentes im physikalischen Speicher, einen Speicherbedarf des Segmentes und eine maximal mögliche Grösse des Segmentes beinhalten.

4. Verfahren nach einem der Ansprüche 1 bis 3, dadurch gekennzeichnet, dass das "Compiler"-Programm (17) alle Segmente des anstehenden Programms (Y_1 bzw. Y_2) in Sektionen zusammenfasst und dass das "Linker"-Programm (20) im Computer (8) der Zentraleinheit (6) die Sektionen anhand einer Spezifikation in einem Speicherbereich plaziert.

5. Verfahren nach Anspruch 4, dadurch gekennzeichnet, dass die Segmentinformationen jeweils eine Sektionszugehörigkeit des Segments beinhalten.

Fig. 1

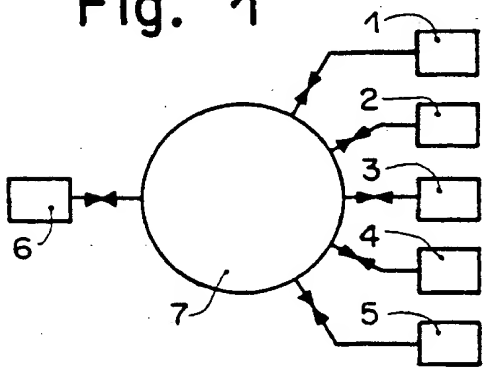


Fig. 2

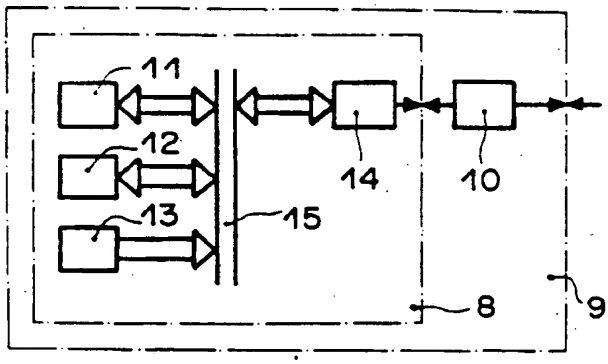


Fig. 3

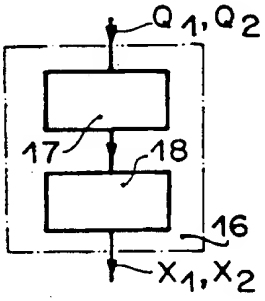


Fig. 4

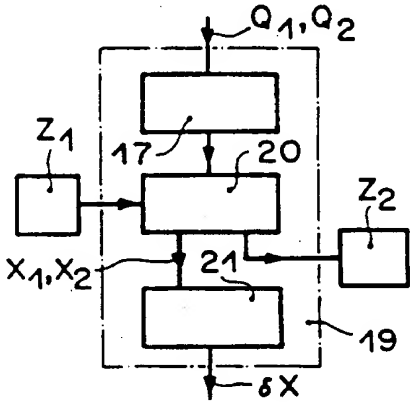


Fig. 5

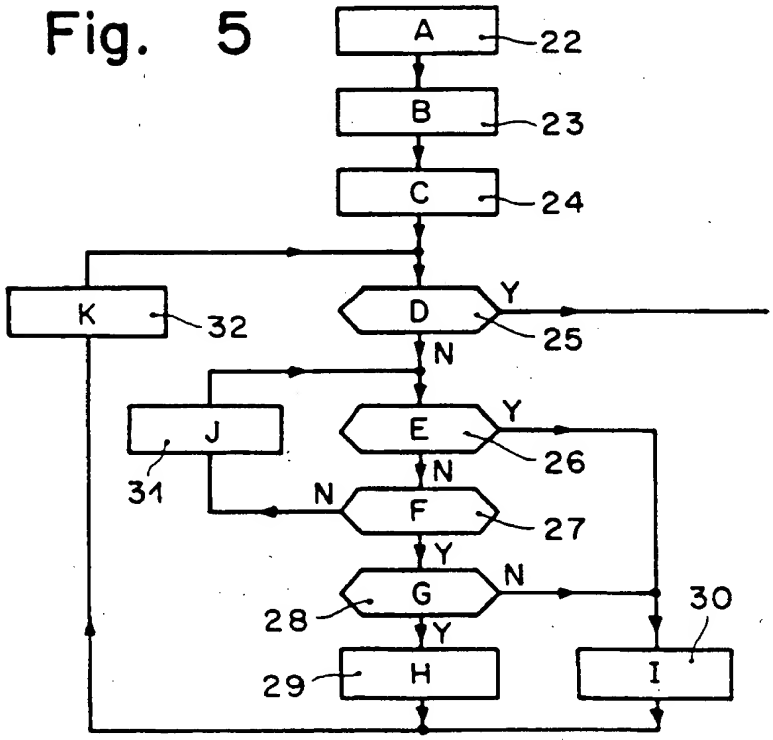


Fig. 6

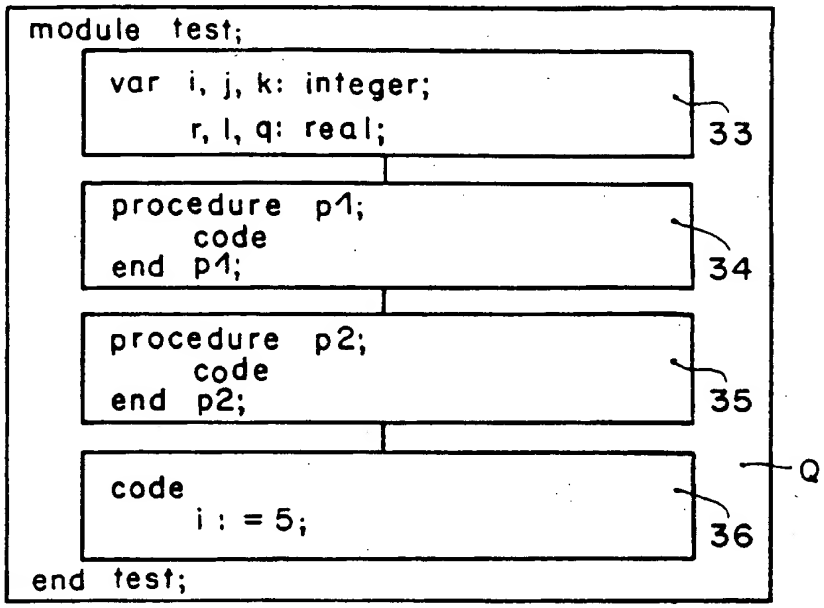


Fig. 7

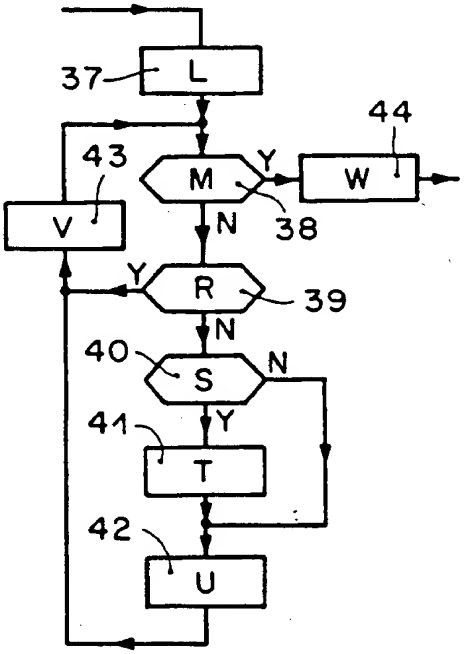
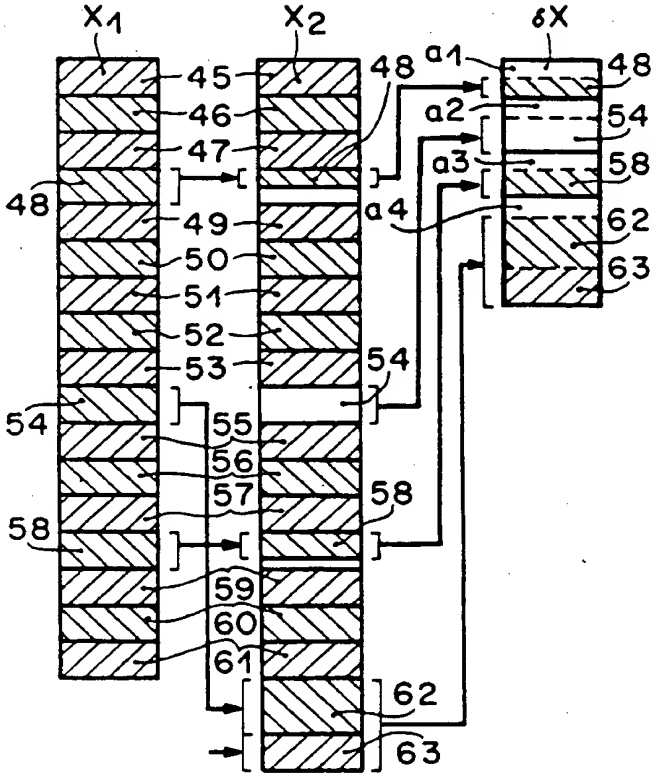


Fig. 8



Europäisches Patentamt

EUROPÄISCHER RECHERCHENBERICHT

Nummer der Anmeldung

EP 91 10 7079

EINSCHLÄGIGE DOKUMENTE

Kategorie	Kennzeichnung des Dokuments mit Angabe, soweit erforderlich, der maßgeblichen Teile	Betrifft Anspruch	KLASSIFIKATION DER ANMELDUNG (Int. Cl.5)
A	EP-A-0 323 707 (WESTINGHOUSE ELECTRIC CORP.) * Zusammenfassung; Figuren 1A,1B; Seite 3, Zeile 1 - Seite 5, Zeile 4 *	1	G 06 F 9/45 G 06 F 9/445
A	EP-A-0 194 822 (SONY CORP.) * Zusammenfassung; Seite 2, Zeilen 15-28; Seite 3, Zeilen 14-23; Seite 4, Zeilen 5-22; Figur 2 *	1	
A	US-A-4 558 413 (SCHMIDT et al.) * Spalte 9, Zeilen 28-59; Spalte 10, Zeilen 4-3 *	1	
A	WO-A-8 301 847 (WESTERN ELECTRIC CO.) * Zusammenfassung; Figuren 1,2; Seite 3, Zeilen 27-38; Seite 10, Zeilen 30-36 *	1	
A	SOFTWARE-PRACTICE AND EXPERIENCE, Band 17, Nr. 7, Juli 1987, Seiten 455-467, Chichester, GB; M.K. CROWE: "Dynamic compilation in the UNIX environment" * Seite 455, Zeile 1 - Seite 456, Absatz 1; Seite 456, letzter Absatz; Seite 457, letzter Absatz - Seite 459, Absatz 1 *	1,4	

RECHERCHIERTE SACHGEBIETE (Int. Cl.5)
G 06 F

Der vorliegende Recherchenbericht wurde für alle Patentansprüche erstellt

Recherchenort	Abschlußdatum der Recherche	Prüfer
Den Haag	26 Juli 91	FONDERSON A.I.

KATEGORIE DER GENANNTEN DOKUMENTE

- X: von besonderer Bedeutung allein betrachtet
- Y: von besonderer Bedeutung in Verbindung mit einer anderen Veröffentlichung derselben Kategorie
- A: technologischer Hintergrund
- O: nichtschriftliche Offenbarung
- P: Zwischenliteratur
- T: der Erfindung zugrunde liegende Theorien oder Grundsätze
- E: älteres Patentdokument, das jedoch erst am oder nach dem Anmeldedatum veröffentlicht worden ist
- D: in der Anmeldung angeführtes Dokument
- L: aus anderen Gründen angeführtes Dokument

& : Mitglied der gleichen Patentfamilie, übereinstimmendes Dokument